

AD-A248 826



2

## DARPA/ONR Quarterly Report

CONTRACT NUMBER:	N00014-91-J-1985
CONTRACTOR:	Duke University
ARPA ORDER NO:	4331714-01/4-6-88
PROGRAM NAME:	N00014
CONTRACT AMOUNT:	696,899.00
EFFECTIVE DATE OF CONTRACT:	July 1, 1991
EXPIRATION DATE OF CONTRACT:	June 30, 1994
PRINCIPAL INVESTIGATOR:	John H. Reif
TELEPHONE NUMBER:	(919) 660-6568
SHORT TITLE OF WORK:	Implementation of Parallel Algorithms
REPORTING PERIOD:	1 Jan. 1992 — 31 March 1992

DTIC  
ELECTE  
APR 20 1992  
S D

The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the U.S. Government.

This document has been approved  
for public release and sale; its  
distribution is unlimited.

92-09660



92 A 11 007

## DESCRIPTION OF PROGRESS--

### **(1) Michael Landis (graduate student), John Reif (PI), and Robert Wagner (Duke faculty): Intermediate Representation for Parallel Implementation**

(1.1) Work is progressing in the design of a new, general intermediate representation for parallel code. We wish to provide a single compilation target that is executable on a wide variety of parallel machines and vector processors. The representation we are designing is an extension of VCODE, due to Blleloch at CMU, which is particularly suitable as a target only for data-parallel languages. VCODE provides a rich set of vector operations, including powerful segmented prefix computation and permutation operations, which run efficiently on a wide range of architectures. However, VCODE provides only sequential control, which limits its application strictly to data-parallel programming. We have extended VCODE to form a new language called AVCODE which has operations for process creation, communication, and synchronization. AVCODE provides an avenue for efficiently targeting a parallel language to a number of different hardware systems through a single intermediate language.

The process model we use is a flexible, hierarchical cactus stack structure in which new processes receive access to a subset of shared memory. This limited access increases remote memory access speed by restricting contention patterns to an individual partition of the shared space. Memory shared between processes is tagged in such a way that communication and synchronization are unified. This method presents a flexible and efficient means of process control, and we will show that the system can be implemented efficiently on varying MIMD architectures.

The tagged memory construct in AVCODE has numerous advantages over other synchronization and communication primitives. First, its simplicity provides a mechanism for parameterising operation cost for high level optimization purposes. Since there is one simple primitive for communication and synchronization, the efficiency of this primitive can be directly measured, enabling the high level language optimizer to make various decisions to optimize process granularity and memory layout. Second, the existence of only one primitive simplifies compilation into AVCODE, since there is no choice among primitives. Our single tagged memory primitive is simple yet general enough such that many synchronization and communication problems can be solved efficiently. We plan to use this new representation as a target for our parallel prototyping language PROTEUS which is currently being implemented.

(1.2) We are in the early stages of investigating methods for optimizing data representations for parallel machines. Given a sequence of operations on a data structure, the choice of representation for that structure has a great impact on program performance. On parallel machines, many issues have to be considered, such as the choice of representation, representation layout, and appropriate times for data structure representation conversion. We are beginning to investigate these optimization issues, especially with regards to set and sequence operations. They will have immediate relevance for the development and implementation of PROTEUS.

### **(2) Peter Mills (Research Associate) with John Reif: Semiautomatic Techniques for Refining High-Level Parallel Programs into Lower-Level Data-Parallel Code**

We have completed an initial investigation into semi-automatic techniques for refining high-level parallel programs into lower-level data-parallel code executable on our common implementation platform AVCODE. Such refinement techniques targeting intermediate broad-spectrum parallel languages form an important component of a methodology for semi-automatically deriving implementations of parallel algorithms executable on families of parallel machines. To validate our approach, we have derived an implementation of a parallel algorithm for potential field calculation

Statement A per telecon  
Dr. Ralph Wachter ONR/Code 1133  
Arlington, VA 22217-5000

NWW 4/17/92

A-1

(N-body problem) which is executable on a wide family of SIMD parallel machines. Our current research efforts are to further develop refinement techniques targeting MIMD, SIMD, and partitionable mixed-mode architectures, and to demonstrate the viability of the refinement techniques and AVCODE by extending our experiment in N-body simulation to more complex algorithms suitable for refinement and implementation on asynchronous collections of SIMD machines. Our experimental implementations of N-body algorithms will form a concrete vehicle for testing not only AVCODE but also other semi-automatic implementation techniques dealing with load balancing and mapping to networks such as hypercubes.

#### Recent Accomplishments:

We have realized refinement techniques which can transform parallel algorithms, for example expressed in the high-level set-theoretic language Proteus, into a form compatible with data-parallel intermediate languages. The refinement involves transforming nested sequence iterators, corresponding to nested parallelism, into a flattened form that can be implemented directly under the SIMD execution model. Other transformations, such as flattening data structures, are also performed to accommodate limitations of the targeted vector model. The transformed program is then in a form compatible with a subset of the common implementation platform AVCODE, which allows it to be executed on a wide range of machines.

We validated the approach by expressing in Proteus the naive algorithm for potential field calculation, refining the program to data-parallel form, translating the result to CVL vector language—an existing implemented broad-spectrum language of which AVCODE is an extension—and then executing the code on a MasPar MP-1. The transformations, although initially performed manually, were successfully automated for this problem using the Kestrel Interactive Development System (KIDS). This promising avenue of semi-automatic transformation uses algebraic laws recognizing a more abstract class of vector operators, such as Cartesian products, than previous data-parallel transformation techniques.

#### Ongoing Work:

- (2.1) Further development of refinement techniques targeting subsets of AVCODE oriented towards MIMD, SIMD, and partitionable mixed-mode architectures.
- (2.2) Demonstration of viability of these techniques through concrete implementations of N-body algorithms, specifically clustering and fast multipole methods, targeting asynchronous collections of SIMD machines.
- (2.3) Extending models of parallel computation with real-time properties, such as processor rates, in order to support timing analysis. Current models, variants of the PRAM such as the APRAM and HPRAM, only accommodate asynchrony of control or hierarchy of control and communication.

### **(3) Peter Su (postdoc) and John Reif: Implementations of Parallel Algorithms in Computational Geometry**

We have been working on the implementational aspects of parallel algorithms. Specifically, we have been studying parallel algorithms for constructing Voronoi Diagrams and related problems. Our interest in this study is not only to build effective algorithms for these problems, but also to consider the kinds of tools that make such work easier and more effective.

Our work has been broken up into three stages:

- a) Study the theory and practice of conventional algorithms for this problem.
- b) Study the current body of theoretical work on parallel algorithms for this problem.

- c) Using the knowledge gained in (a) and (b), design and implement parallel algorithms for this problem on several machines. Then study the performance of these algorithm and how well the theoretical results match the behavior of the implementation.

We have been actively working on stages (a) and (b) for the last few months and we are now ready to move on to stage (c). The study of practical sequential algorithms has been especially helpful in the pursuit of simple and efficient parallel algorithms, since they provide a good set of ideas to extend and refine in a parallel setting.

Using the experience that we gain from this work, we are also investigating and planning tools that could aid the programmer in implementing effective parallel algorithms. Since many parallel algorithms, especially in computational geometry, have similar structure, one could imagine a tool for reasoning about abstract classes of algorithms. In particular, such a system could aid the programmer in tuning performance parameters for specific machines based on architectural characteristics such as global memory bandwidth and latency, processor speed, local memory size, and so on. Also, more basic tools for doing visualization and performance analysis are needed to help the programmer to effective experimental analysis of his implementations. Tools for profiling, algorithm animation, simulation and data analysis would all be extremely useful in these settings. At this point, there are no such tools widely available to the research community.

#### **(4) Shenfeng Chen with John Reif: Parallel Sort Implementation**

Summary:

The fastest known sort is a parallel implementation of radix sort in a CRAY, due to CMU's Guy Blelloch. The current sorting algorithms on parallel machines like Cray and CM-2 use radix and bucket sort. But they are not taking advantage of possible distribution of the input keys. We are developing an algorithm using data compression to achieve a fast parallel algorithm which takes this advantage. We expect the new algorithm to beat the previous fastest sort by a few factors. We are working to implement this new parallel sorting algorithm on various parallel machines.

Details:

Radix sort is very efficient when the input keys can be viewed as bits. But the basic radix sort is not distribution based so it needs to look up all digits.

Our approach is to find the structure (distribution) of the input. This is achieved by building a prefix tree after sampling from the original set. Then the prefix tree is trimmed to have all leaves denoting a set of keys with essentially the equal size. A probability analysis shows that the largest set can be bound within a constant of the average size. Building a prefix tree uses techniques of data compression.

Once a good prefix tree is built, basic sorting algorithm is applied for the original set indexing to the tree. Three hash tables are used for indexing purpose. The index length of the first one is the average depth of leaves of the prefix tree. Another two is of the longest depth and shortest depth, respectively. All keys are assigned to each bucket after indexing. Then the standard sorting is used for sorting keys within each bucket separately.

The main advantage of this algorithm is to make the indexing much fast because not all bits are compared when indexing to buckets.

Our first step, now in progress, is to build an algorithm based on this idea. Then we will do an implementation on the Cray to compare our results with the others.

**(5) Deganit Armon (A.B.D.) with John Reif: Parallel Implementation of Nested Dissection.**

**Summary:**

We worked on parallel implementation of nested dissection, a numerical method for solving large sparse systems of linear equations, showing three improvements to the known algorithms and implementations. These include a reduction in the memory requirements of the algorithm, a widening of the class of problems solvable with parallel nested dissection (PND) on a mesh-connected processor array, and a reduction in the asymptotic time bounds of PND.

**Details:**

Nested dissection is a method for solving sparse linear systems of equations by exploiting the graph structure underlying the input matrix.

One of the problems with the known implementations of PND is the large storage requirements of the algorithm; these limit the size of problem for which PND is useful. We show that it is possible to significantly reduce the storage used by an implementation on a processor array to a constant factor of the size of the input matrix. This improvement can be added without affecting the time bounds of the algorithm. We are currently working on an actual implementation.

Using load balancing techniques, we show that PND can be used to solve a larger class of problems on a mesh-connected processor array. In particular, we can use PND to solve any system of equations whose underlying graph is of bounded degree.

We improve on the results of Pan and Reif, who showed that PND can be implemented in  $O(\log^3 n)$  time. By taking into account the fact that processors are idle during later stages of the algorithm, several stages can be grouped and performed together to achieve an  $O(\log^2 n)$  time algorithm for hypercubes.

An area of ongoing investigation is the implementation of PND on various parallel models. One possible such model that we are looking at is the hierarchical model of parallel memory proposed by Heywood, which may be closer to existing parallel architecture.

**(6) Michael Landis (graduate student) with Robert Wagner (faculty): A Method for Deriving Systolic Algorithms**

We have completed our work in the investigation of how to map context-free grammar recognition onto systolic arrays, and a paper entitled "A Method for Deriving Systolic Algorithms" has been submitted to the IEEE Transactions on Computers.

Currently, work is proceeding steadily along three directions.

(6.1) We are developing ways of evaluating uniform expressions in near minimum parallel time on processor arrays. The problem has been solved for two-dimensional arrays, and we are currently working on higher order generalizations. The method is very practical, and has immediate usefulness for performing reduction operations on sets or sequences. A paper describing the work in two dimensions has been submitted to the Journal of the ACM; an abstract of this paper follows.

### ABSTRACT

Consider an array of Processing Elements [PEs], connected by a 2-dimensional grid network, and holding at most one operand of an expression in each PE. Suppose that each PE is allowed, in any one parallel step, to receive one item of data from any of its 4 immediate neighbors, and to transmit one datum, as well. How can an associative operator, such as addition, combine all the operands, using as little time for communication as possible? An expression using such a single operator is termed a uniform expression. When the total number of communication links used is the measure of goodness, this problem becomes a Steiner Tree problem, in the Manhattan Distance metric. When the measure is minimizing the parallel time to completion, a method for solving this problem is given which is optimal to within an additive constant of 2 time-steps. The method has applications when the operands are matrices, spread over an array of PE's, as well. Some lower bounds for this problem, in more general networks, are also proven.

#### (6.2) Details:

We have developed a new method for mapping algorithms into parallel architectures. This new method works very well for a class of dynamic programming problems, including CFG recognition.

In our work, we define an instance of an algorithm may be represented as a DAG. Mapping the algorithm onto a parallel architecture then is analogous to scheduling the nodes in the DAG, i.e. to assign to each computation in the algorithm a time and a location that is consistent with the architecture. To ease this task, we have proven that a restricted form of recurrence equation, called a quasi-uniform recurrence equation (QURE), is computationally equivalent to a class of systolic array architectures. The class of QUREs is a broad class, the main restriction being that the recurrence must exhibit a finite set of dependency vectors over all computations. Expressing an algorithm as a recurrence equation, the problem of mapping the algorithm onto a systolic array becomes the problem of translating the recurrence equation into QURE form. It is also very easy to translate between the DAG representation and the recurrence relation.

We have noted that intermediate results and elementary data are generally used again and again in many algorithms, especially dynamic programming algorithms. We have also noted that the task of moving these data and results around is the most constraining factor for most architectures. We have determined that the number of copies that an architecture can make of an operand in one time step to some degree characterizes the communication constraints of most architectures.

In our method, we take the original DAG and label its arcs and nodes to meet two constraints: The number of copies of each computation that can exist in each timestep, and the number of data that each computation can receive in each timestep. To implement these constraints, we proceed through several steps:

- a) Derive a computational indexing scheme for the algorithm. The index enumerates all of the intermediate results in the algorithm such that each result  $C(i)$  is dependent on results  $C(j)$  where  $i > j$ . Each  $C(i)$  is called a computation point.
- b) Build a time-availability table  $t(i,j)$  and vector  $T(j)$  where each  $t(i,j)$  is filled with the time that each computation point  $C(i)$  is available for use in the evaluation of  $C(j)$ , and  $T(j)$  is the minimum time at which  $C(j)$  can be computed.

In building this table, we also make use of further important constraint. We note that certain operands need to be used with other operands. We call this constraint the simultaneous availability constraint, since one operand must become available with another. This constraint directly affects the values in the table  $t(i,j)$ .

- c) We then augment this table with information about the source through which each datum may be passed.
- d) We then invert this table to determine what data is passed on which computation points at each time.

Current investigation on the extension of our method mostly involves step (a). Deriving the computational indexing scheme is analogous to solving a specific graph layout problem. In order to achieve a QURE as the result of our method, the nodes in the DAG must be labeled in a multi-dimensional coordinate system such that the computation of each node only depends upon nodes that are a fixed distance away. We are currently investigating a shortest-path method to perform this indexing, which promises to work for a broad class of algorithms—not just dynamic programming.

### (6.3) Debugging Facilities:

We proposed initial set of debugging facilities for PROTEUS, based on Paul Hudak's notion that "debugging" operations can be embedded in the language proper

### (6.4) Data movement

In looking at the problems of distributing collection-oriented operations and collections across many MIMD processors, the question of data-communication cost comes up.

Suppose PE's are connected in a 2-D grid, with the property that any PE can receive a datum from any *one* neighbor at a given time-step. (Other network models, and PE communication behavior schemes are also possible.) In this setting, ignoring operation costs, we've studied the problem

of computing  $\sum v_i$ , where each  $v_i$  is originally located on a different PE. The goal is to minimize parallel communication time. We have written a TR that solves the problem within 2 steps of optimality regardless of the initial placement of the operands in the grid.

This sort of study opens an area of research, delimited by choosing different combinations of assumptions about PE behavior, network topology, and problems to be solved. A systematic study of these questions, oriented toward problems which are communication-intensive and of interest to people developing packages like LINPAK, seems in order.

## (7) Prokash Sinha with John Reif: Randomized Parallel Algorithms for Min Cost Paths

### Summary:

We have completed our initial investigation to derive randomized parallel algorithms for Min Cost Paths in a Graph of High Diameter. Present accomplishment is a randomized sequential algorithm with an order of magnitude performance gain for some dense graphs. Currently we are in the process of submitting this finding in technical journals and conferences. Our next phase of work is to give a similar algorithm for a unit cost machine. Then we will extend the approach to other problems. Our current research effort is to extend the techniques of Flajolet and Karp to Develop

techniques and tools for timing analysis of algorithms. This effort is to derive tools for semiautomatic randomized analysis.

We have written the following paper: "A Randomized Algorithm for Min Cost Paths in a Graph of High Diameter: Extended Abstract" (J. Reif and P. Sinha), 1992.

**(8) Duke High Performance Computing Seminar:**

Participants--Professors Robert Wagner, Donald Loveland, Gopalan Nadathur, Donald Rose, Paul Lankron, and John Reif; Deganit Armon, Shenfeng Chen, Michael Landis, Peter Mills, Peter Su, Steve Tate, Akitoshi Yoshida.

**(9) Researchers supported (other than PI):**

Salman Azhar, graduate student  
Mike Landis, graduate student  
Peter Mills, post-doc  
Robert Wagner, professor  
Akitoshi Yoshida, graduate student

**(10) Degrees awarded:**

Sandeep Sen received his Ph.D. from Duke and was here as a post-doc. Steve Tate and Lars Nyland received their Ph.D.s in January 1991 under Reif. Tate is remaining at Duke as a post-doc.

**(11) Papers**

- (1) Strong k-connectivity in Digraphs and Random Digraphs (with P. Spirakis), accepted to *Journal of Algorithmica*, 1992.
- (2) Probabilistic Parallel Prefix Computation, *13th Annual International Conference on Parallel Processing*, Michigan, 1984; also Harvard University TR-08-83. Accepted for publication in *Computers and Mathematics with Applications*, 1992.
- (3) Efficient VLSI Fault Simulation, Technical Report TR-03-85, CRCT, Harvard University. Also to appear in *Computers and Mathematics with Applications*, 1992.
- (4) Optimal Parallel Algorithms in Computational Geometry (with S. Sen). *1987 International Conference on Parallel Processing*, University Park, PA, August 1987. Also in *Algorithmica*, vol. 1, pp 91-117, January 1992.
- (5) On Threshold Circuits and Polynomial Computation. *2nd Structure in Complexity Theory Conference*, Ithaca, NY, June 1987. Also revised as "On Threshold Circuits and Efficient, Constant Depth Polynomial Computation" (with S. Tate), August 1988. Accepted for publication in *SIAM Journal of Computing*, 1992
- (6) Nested Annealing: A Provable Improvement to Simulated Annealing (with S. Rajasekaran), MCNC Technical Report TR-88-19; presented at *Workshop on Applications of Combinatorics and Graph Theory to Computer Science*, Institute for Mathematics and its Applications, University of Minnesota, December 13-17, 1987; also appeared in the *15th International Colloquium on Automata, Languages and Programming*, Tampere, Finland, July 1988. Accepted for publication in *Journal of Theoretical Computer Science*, November 1992.
- (7) Polling: A New Randomized Sampling Technique for Computational Geometry (with S. Sen). *21st Annual ACM Symposium on Theory of Computing*, pp. 394-404, Seattle, WA, May 1989. Also revised as "Optimal Parallel Algorithms for 3 Dimensional Convex Hulls and Related Problems", *SIAM Journal of Computing*, vol. 21, no. 3, June 1992.



## (11) Papers (continued)

- (8) On Applications of Crypto-complexity to Analyzing Efficiency of Capital Markets (with S. Azhar). Submitted for journal publication, 1992.
- (9) Continuous Alternation, (with S. Tate), to appear in a special issue of *Journal of Algorithmica*, edited by B. Donald, 1992.
- (10) Optical Expanders with Applications in Optical Computing (with A. Yoshida). Submitted for journal publication, August 1989. Submitted to *Journal of Applied Optics*, 1991.
- (11) Error Resilient One-way Dynamic Communication, (with J. A. Storer). Submitted for journal publication, October 1990.
- (12) Prototyping Parallel and Distributed Programs in Proteus, (with P. H. Mills, L.S. Nyland, J.F. Prins, R.A. Wagner). *Third IEEE Symposium on Parallel and Distributed Processing*, Dallas, TX, December 1991
- (13) A Massively Parallel VLSI Compression System using a Compact Dictionary, (with J.A. Storer and T. Markas), *Proceedings of IEEE Workshops on VLSI & Signal Processing*, 1990, San Diego, CA. Published as "A Massively Parallel VLSI Compression System Using a Compact Dictionary", *VLSI Signal Processing*, no. 4, 1990 (edited by H.S. Moscovitz and K. Yao and R. Jain), IEEE Press, 1990, New York, NY, pp. 329-338.
- (14) Towards Randomized Strongly Polynomial Algorithms for Linear Programming (with S. Krishnan), Submitted for journal publication, 1990. Duke University Technical Report CS-1991-18.
- (15) Decreasing the Precision of Linear Algebra Computations by Using Compact Multigrid and Backward Interval Analysis (with V. Pan). *4th SIAM Conference in Applied Linear Algebra*, Minneapolis, MN, September 1991.
- (16) Fast Computations of Vector Quantization Algorithms (with T. Markas), NASA Technical Report TR-91-58, 1991.
- (17) Image Compression Methods with Distortion Controlled Capabilities (with T. Markas). *Proceedings of Data Compression Conference (DCC 91)*, Snowbird, UT, pp. 93-102, April 1991. Revised (with T. Markas) as "Quad Tree Structures for Image Compression Applications", special issue of *Journal of Information Processing and Management*, 1992.
- (18) Algebraic Methods for Testing the k-Vertex Connectivity of Directed Graphs, (with J. Cheriyan), *3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, 1992.
- (19) Optical Techniques for Image Compression (with A. Yoshida). *2nd Annual IEEE Data Compression Conference*, Salt Lake City, UT, March 1992.
- (20) The Complexity of N-body Simulation (with S. Tate). Submitted for journal publication, 1992.
- (21) Searching in an Unknown Environment (with M. Kao and S. Tate). Submitted for journal publication, 1992.
- (22) Fully Dynamic Graph Connectivity in Logarithmics Expected Time (with P. Spirakis and M. Yung). Submitted for journal publication, 1992.
- (23) On Parallel Implementations and Experimentations of Lossless Data Compression Algorithms (with T. Markas). Submitted to Data Compression Conference 1992.
- (24) Memory-Shared Parallel Architectures for Vector Quantization Algorithms (with T. Markas). Submitted for journal publication, 1992.
- (25) A Randomized Algorithm for Min Cost Paths in a Graph of High Diameter: Extended Abstract (with P. Sinha), 1992.
- (26) Prototyping N-body Simulation in Proteus (with P. Mills, L. Nyland, and J. Prins). To appear in *Proceedings of the Sixth International Parallel Processing Symposium*, Beverly Hills, CA, March 23-26, 1992.
- (27) An Optimal Space and Efficient Parallel Nested Dissection Algorithm (with D. Armon). Submitted for journal publication, 1992.

**SUPPLEMENTARY**

**INFORMATION**

A 348856

## ERRATA

### DARPA/ONR Quarterly Report — Addendum

CONTRACT NUMBER:	N00014-91-J-1985
CONTRACTOR:	Duke University
ARPA ORDER NO:	4331714-01/4-6-88
PROGRAM NAME:	N00014
CONTRACT AMOUNT:	696,899.00
EFFECTIVE DATE OF CONTRACT:	July 1, 1991
EXPIRATION DATE OF CONTRACT:	June 30, 1994
PRINCIPAL INVESTIGATOR:	John H. Reif
TELEPHONE NUMBER:	(919) 660-6568
SHORT TITLE OF WORK:	Implementation of Parallel Algorithms
REPORTING PERIOD:	1 Jan. 1992 — 31 March 1992

The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the U.S. Government.

02 4 21 052

92-10232



021 North Building  
Durham, NC 27706  
(919) 660-6568  
Email: reif@cs.duke.edu

14 April 1992

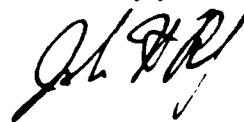
**DARPA/ONR Quarterly Report — Addendum**  
**Contract Number: N00014-91-J-1985**  
**Short Title: Implementation of Parallel Algorithms**  
**Reporting Period: 1 Jan 1992 — 31 March 1992**

We are pleased to report that the following three papers were recently accepted for presentation at the prestigious 4th Annual ACM Symposium on Parallel Algorithms and Architectures, June 29–July 1, 1992, in San Diego, California:

- (1) Implementations of Randomized Sorting on Large Parallel Machines (by W. Hightower, J. Prins, and J. Reif),
- (2) Space and Time Efficient Implementations of Parallel Nested Dissection (by D. Armon and J. Reif),
- (3) Efficient Parallel Algorithms for Computing All Pair Shortest Paths in Directed Graphs (by Y. Han, V. Pan, and J. Reif).

These papers will be published in the symposium proceedings.

Sincerely yours,



John H. Reif  
Professor